## Section I:
## AMENDMENT UNDER 37 CFR §1.121 to the CLAIMS

Claim 1 (currently amended):

A method for marking a processing stack with signatures to indicate which portions of the stack were utilized by [[which]] one or more re-entrant or object-oriented programming software code modules, said method comprising the [[step]] steps of:

inserting stack signing software into one or more re-entrant or object-oriented programming code modules stored in a computer-readable medium;

~~assigning unique module identifier values to a plurality of code modules; and~~

producing one or more executable programs containing one or more executable re-entrant or object oriented programming code modules containing said inserted stack signing software;

upon execution of said executable re-entrant or object-oriented code modules, assigning unique module identifier values to said ~~a plurality of~~ code modules by said stack signing software, said stack signing software preventing module identifiers from having a same value for multiple instances of any re-entered or multiply instantiated code module, and pushing onto [[a]] said processing stack said ~~unique module identifiers~~ stack signatures within stack frames allocated to said code modules.

Claim 2 (currently amended):

The method as set forth in Claim 1 further comprising the steps of:

generating an instance ~~number~~ count for each instantiation of [[a]] executable code module in said stack signature for each object instance dynamically created during runtime of a re-entrant executable code module; and

pushing onto said processing stack said instance count ~~numbers associated with said unique module identifier values~~.

Claim 3 (original):

The method as set forth in Claim 1 further comprising the step of pushing onto said stack an entry/exit indicator associated with said unique module identifier.

Claim 4 (original):

The method as set forth in Claim 1 further comprising the step of inserting stack signature marking software segments into application source code, said insertion step being performed prior to compilation of said application source code.

Claim 5 (original):

The method as set forth in Claim 4 further comprising the step of providing a global control which indicates all application source code modules are to have stack signature marking software segments inserted into them during a given compilation job.

Claim 6 (original):

The method as set forth in Claim 4 further comprising the step of providing a selective control which indicates only certain application source code modules are to have stack signature marking software segments inserted into them during a given compilation job.

Claim 7 (currently amended):

A computer readable medium encoded with software for marking a processing stack with signatures to indicate which portions of the stack were utilized by which application code modules, said software causing a processor to perform the steps of:

inserting stack signing software into one or more re-entrant or object-oriented programming code modules stored in a computer-readable medium;

~~assigning unique module identifier values to a plurality of code modules; and~~

producing one or more executable programs containing one or more executable re-entrant or object oriented programming code modules containing said inserted stack signing software;

upon execution of said executable re-entrant or object-oriented code modules, assigning unique module identifier values to said ~~a plurality of~~ code modules by said stack signing software, said stack signing software preventing module identifiers from having a same value for multiple instances of any re-entered or multiply instantiated code module, and pushing onto [[a]] said processing stack said ~~unique module identifiers~~ stack signatures within stack frames allocated to said code modules.

Claim 8 (original):

The computer readable medium as set forth in Claim 7 further comprising software to perform the steps of:

generating an instance number for each instantiation of a code module; and

pushing onto said processing stack said instance numbers associated with said unique module identifier values.

Claim 9 (original):

The computer readable medium as set forth in Claim 7 further comprising software for performing the step of pushing onto said stack an entry/exit indicator associated with said unique module identifier.

Claim 10 (original):

The computer readable medium as set forth in Claim 7 further comprising software for performing the step of inserting stack signature marking software segments into application source code, said insertion step being performed prior to compilation of said application source code.

Claim 11 (original):

The computer readable medium as set forth in Claim 10 further comprising software for performing the step of providing a global control which indicates all application source code modules are to have stack signature marking software segments inserted into them during a given compilation job.

Claim 12 (original):

The computer readable medium as set forth in Claim 10 further comprising software for performing the step of providing a selective control which indicates only certain application source code modules are to have stack signature marking software segments inserted into them during a given compilation job.

Claim 13 (currently amended):

A system for inserting stack signature marking code segments into application software modules prior to compilation, said system cooperating with a compiler and comprising:

a control means operable by a user to indicate whether or not to insert stack signature marking code segments into application software modules; [[and]]

a code insertion means which, responsive to the operation of the control means, searches for entry points and exits points in application software modules and inserts stack signature marking code segments following each entry point and prior to each exit point into said application software modules[[.]] ;

a compiler means for producing one or more executable programs containing one or more executable re-entrant or object oriented programming code modules containing said inserted stack signing software; and

a debugger means configured to, upon execution of said executable re-entrant or object-oriented code modules, assign unique module identifier values to said code modules by said stack signing software, said stack signing software preventing module identifiers from having a same value for multiple instances of any re-entered or multiply instantiated code module, and to push onto said processing stack said stack signatures.

Claim 14 (currently amended):

The system of Claim [[14]] 13 wherein said control means comprises a global control means for indicating insertion of stack signature marking code segments are to be inserted into all application software modules to be compiled.

Claim 15 (currently amended):

The system of Claim [[15]] 13 wherein said control means comprises a selective control means for indicating specific applications software modules or groups of application software modules into which stack signature marking code segments are to be inserted.

Claim 16 (new):

The method as set forth in Claim 1 further comprising encrypting at least a portion of said stack signature.

Claim 17 (new):

The method as set forth in Claim 2 further comprising encrypting said instance number in said stack signature.

Claim 18 (new):

The method as set forth in Claim 1 wherein said step of pushing stack signatures onto said processing stack comprises:

generating a pseudo-random identifier for each object instance dynamically created during runtime of a re-entrant executable code module; and

including said pseudo-random identifier in said stack signature pushed onto said processing stack.